



US Patent &amp; Trademark Office

[Subscribe](#) (Full Service) [Register](#) (Limited Service, Free) [Login](#)Search: ☐ The Guide ☒ The ACM Digital Library

graph AND nodes AND edges AND edit AND direction AND fold



THE ACM DIGITAL LIBRARY

[Incident report](#)

Terms used

**graph** AND **nodes** AND **edges** AND **edit** AND **direction** AND **folding**

Found 36,411 of 111,550

Sort results  
by

relevance

Display  
results

expanded form

[Save results to a Binder](#) [Search Tips](#)☐ Open results in a new  
window[Try an Advanced Search](#)[Try this search in The ACM Guide](#)

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐**1** [Incremental global reoptimization of programs](#)

Lori L. Pollock, Mary Lou Soffa

April 1992 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,

Volume 14 Issue 2

Full text available: [pdf\(1.88 MB\)](#)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Although optimizing compilers have been quite successful in producing excellent code, two factors that limit their usefulness are the accompanying long compilation times and the lack of good symbolic debuggers for optimized code. One approach to attaining faster recompilations is to reduce the redundant analysis that is performed for optimization in response to edits, and in particular, small maintenance changes, without affecting the quality of the generated code. Although modular program ...

**Keywords:** compiler optimization, incremental data flow analysis, incremental reoptimization, optimization dependencies

**2** [Synthesis of bent sheet metal parts from design features](#)

Roger Bush, Carlo Sèquin

June 1999 **Proceedings of the fifth ACM symposium on Solid modeling and applications**Full text available: [pdf\(1.51 MB\)](#)Additional Information: [full citation](#), [references](#), [index terms](#)

**Keywords:** automated design, sheet metal part design

**3** [Pleasure: a computer program for simple/multiple constrained unconstrained folding of programmable logic arrays](#)


G. DeMichelli, A. Sangiovanni-Vincentelli

June 1988 **Papers on Twenty-five years of electronic design automation**Full text available: [pdf\(1.04 MB\)](#)Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)**4**[Incremental Context-Dependent Analysis for Language-Based Editors](#)



- Thomas Reps, Tim Teitelbaum, Alan Demers

July 1983 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,  
Volume 5 Issue 3

Full text available:  [pdf\(1.76 MB\)](#)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

##### 5 PLEASURE: a computer program for simple/multiple constrained/unconstrained folding of Programmable Logic Arrays

Giovanni De Micheli, Alberto Sangiovanni-Vincentelli

June 1983 **Proceedings of the twentieth design automation conference on Design automation**

Full text available:  [pdf\(877.59 KB\)](#)

Additional Information: [full citation](#), [references](#), [index terms](#)

##### 6 The program dependence graph and its use in optimization

Jeanne Ferrante, Karl J. Ottenstein, Joe D. Warren

July 1987 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,  
Volume 9 Issue 3

Full text available:  [pdf\(2.51 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

In this paper we present an intermediate program representation, called the program dependence graph (PDG), that makes explicit both the data and control dependences for each operation in a program. Data dependences have been used to represent only the relevant data flow relationships of a program. Control dependences are introduced to analogously represent only the essential control flow relationships of a program. Control dependences are derived from the ...

##### 7 Graph rewrite systems for program optimization

Uwe Assmann

July 2000 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,  
Volume 22 Issue 4

Full text available:  [pdf\(571.22 KB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Graph rewrite systems can be used to specify and generate program optimizations. For termination of the systems several rule-based criteria are developed, defining exhaustive graph rewrite systems. For nondeterministic systems stratification is introduced which automatically selects single normal forms. To illustrate how far the methodology reaches, parts of the lazy code motion optimization are specified. The resulting graph rewrite system classes can be e ...

**Keywords:** compiler generators, graph rewrite systems, program analysis, program optimization, program transformation, specification, stratification, very high-level languages, visual programming

##### 8 On the complexity of protein folding (extended abstract)

Pierluigi Crescenzi, Deborah Goldman, Christos Papadimitriou, Antonio Piccolboni, Mihalis Yannakakis

May 1998 **Proceedings of the thirtieth annual ACM symposium on Theory of computing**

Full text available:  [pdf\(748.15 KB\)](#)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)



9 Incremental evaluation for attribute grammars with application to syntax-directed editors

Alan Demers, Thomas Reps, Tim Teitelbaum

January 1981 **Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Full text available:  pdf(999.02 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

A syntax-directed editor is a tool for structured program development. Such an editor can enforce syntactic correctness incrementally by restricting editing operations to legitimate modifications of the program's context-free derivation tree. However, not all language features can be described by the context-free formalism. To build editors that enforce non-context-free correctness, a more powerful specification technique is needed. In this paper we discuss the advantages of attribute grammars a ...

10 Annotation-directed run-time specialization in C

Brian Grant, Markus Mock, Matthai Philipose, Craig Chambers, Susan J. Eggers

December 1997 **ACM SIGPLAN Notices , Proceedings of the 1997 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation**, Volume 32 Issue 12

Full text available:  pdf(1.99 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

We present the design of a dynamic compilation system for C. Directed by a few declarative user annotations specifying where and on what dynamic compilation is to take place, a binding time analysis computes the set of run-time constants at each program point in each annotated procedure's control flow graph; the analysis supports program-point-specific polyvariant division and specialization. The analysis results guide the construction of a specialized run-time specialization for each dynamically c ...

11 On the learnability and usage of acyclic probabilistic finite automata

Dana Ron, Yoram Singer, Naftali Tishby

July 1995 **Proceedings of the eighth annual conference on Computational learning theory**

Full text available:  pdf(1.14 MB) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

12 A simple graph-based intermediate representation

Cliff Click, Michael Paleczny

March 1995 **ACM SIGPLAN Notices , Papers from the 1995 ACM SIGPLAN workshop on Intermediate representations**, Volume 30 Issue 3


Full text available:  pdf(1.39 MB) Additional Information: [full citation](#), [abstract](#), [index terms](#)

We present a graph-based intermediate representation (IR) with simple semantics and a low-memory-cost C++ implementation. The IR uses a directed graph with labeled vertices and ordered inputs but unordered outputs. Vertices are labeled with opcodes, edges are unlabeled. We represent the CFG and basic blocks with the same vertex and edge structures. Each opcode is defined by a C++ class that encapsulates opcode-specific data and behavior. We use inheritance to abstract common opcode behavior ...

13 Space and time-efficient memory layout for multiple inheritance

Peter F. Sweeney, Joseph (Yossi) Gil

October 1999 **ACM SIGPLAN Notices , Proceedings of the 1999 ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10

Full text available:  pdf(2.30 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)



Traditional implementations of multiple inheritance bring about not only an overhead in terms of run-time but also a significant increase in object space. For example, the number of compiler-generated fields in a certain object can be as large as quadratic in the number of its subobjects. The problem of efficient object layout is compounded by the need to support two different semantics of multiple inheritance: shared, in which a base class inherited along distinct ...

**14 Geometric modeling of solid objects by using a face adjacency graph representation**

Silvia Ansaldi, Leila De Floriani, Bianca Falcidieno

July 1985 **ACM SIGGRAPH Computer Graphics , Proceedings of the 12th annual conference on Computer graphics and interactive techniques**, Volume 19 Issue 3

Full text available:  [pdf\(715.25 KB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

**15 Analysis and transformation in the ParaScope editor**

Ken Kennedy, Kathryn S. McKinley, Chau-Wen Tseng

June 1991 **Proceedings of the 5th international conference on Supercomputing**

Full text available:  [pdf\(1.71 MB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

**16 Protein folding in the hydrophobic-hydrophilic (HP) is NP-complete**

Bonnie Berger, Tom Leighton


March 1998 **Proceedings of the second annual international conference on Computational molecular biology**

Full text available:  [pdf\(1.27 MB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

**17 Optimization of functional programs by grammar thinning**

Adam Webber

March 1995 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 17 Issue 2

Full text available:  [pdf\(2.43 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

We describe a new technique for optimizing first-order functional programs. Programs are represented as graph grammars, and optimization proceeds by counterexample: when a graph generated by the grammar is found to contain an unnecessary computation, the optimizer attempts to reformulates the grammar so that it never again generates any graph that contains that counterexample. This kind of program reformulation corresponds to an interesting problem on context-free grammars. Our reformulation ...

**Keywords:** functional languages, graph grammars, optimization

**18 Motorcycle graphs and straight skeletons**

Siu-Wing Cheng, Antoine Vigneron

January 2002 **Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms**

Full text available:  [pdf\(891.38 KB\)](#) Additional Information: [full citation](#), [abstract](#)


We present a new algorithm to compute a motorcycle graph. It runs in  $O(n\sqrt{n} \log n)$  time when  $n$  is the size of the input. We give a new characterization of the straight skeleton of a polygon possibly with holes. For a simple polygon, we show that it yields a randomized



- algorithm that reduces the straight skeleton computation to a motorcycle graph computation in  $O(n \log^2 n)$  time. Combining these results, we can compute the str ...

**19** Stochastic roadmap simulation: an efficient representation and algorithm for analyzing molecular motion

Mehmet Serkan Apaydin, Douglas L. Brutlag, Carlos Guestrin, David Hsu, Jean-Claude Latombe  
April 2002 **Proceedings of the sixth annual international conference on Computational biology**

Full text available:  pdf(2.03 MB)


Additional Information: [full citation](#), [abstract](#), [references](#)

Classic techniques for simulating molecular motion, such as the Monte Carlo and molecular dynamics methods, generate individual motion pathways one at a time and spend most of their time trying to escape from the local minima of the energy landscape of a molecule. Their high computational cost prevents them from being used to analyze many pathways. We introduce Stochastic Roadmap Simulation (SRS), a new approach for exploring the kinetics of molecular motion by simultaneously examining multip ...

**20** Type fixpoints: iteration vs. recursion

Zdzisław Szpawski, Paweł Urzyczyn

September 1999 **ACM SIGPLAN Notices , Proceedings of the fourth ACM SIGPLAN international conference on Functional programming**, Volume 34 Issue 9

Full text available:  pdf(1.39 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Positive recursive (fixpoint) types can be added to the polymorphic (Church-style) lambda calculus  $\lambda_{\rightarrow}^2$  (System F) in several different ways, depending on the choice of the elimination operator. We compare several such definitions and we show that they fall into two equivalence classes with respect to mutual interpretability by means of beta-eta reductions. Elimination operators for fixpoint types are thus classified as either "iterators" or "recursors". This classification has a ...

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2003 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:



[Adobe Acrobat](#)



[QuickTime](#)



[Windows Media Player](#)



[Real Player](#)